PROGRAMACIÓN PARA SISTEMAS

TEMA 2

Unix: Shell y Scripts

Unix: Shell y Scripts

INTRODUCCIÓN

- LOS PUNTOS QUE SE TRATARÁN SON LOS SIGUIENTES:
  - GRUPOS DE COMANDOS Y FUNCIONES
  - COMANDOS SÍNCRONOS Y ASÍNCRONOS
  - DEFINICIÓN DE VARIABLES
  - EXPANSIÓN DE VARIABLES
  - SEÑALES / INTERRUPCIONES
  - SHELLS Y SUBSHELLS
  - ENTORNO DE EJECUCIÓN

BIBLIOGRAFÍA RECOMENDADA:

- ***The UNIX Programming Environment, B. Kernighan, R. Pike***
- ***BASH Cookbook, C. Albing, JP. Vossen, C. Newham***

Unix: Shell y Scripts

GRUPOS DE COMANDOS Y FUNCIONES

- SE PUEDEN DEFINIR GRUPOS DE COMANDOS COMPUESTOS DE VARIOS COMANDOS ENCERRADOS POR PARÉNTESIS O LLAVES Y SEPARADOS ENTRE SÍ POR (;), *pipes* (|) O *newline* :

  (*comando1; comando2; …)* *{ comando1; comando2; … }*

- IGUALMENTE, SE PUEDEN DEFINIR FUNCIONES:

```
nombre_funcion()
 {
        comando1
        comando2
          ….
 }
```

QUE SE EJECUTAN COMO SI SE TRATARA DE UN COMANDO

Unix: Shell y Scripts

GRUPOS DE COMANDOS Y FUNCIONES

- IGUALMENTE, SE PUEDEN DEFINIR LISTAS DE COMANDOS ENLAZADOS CON LOS OPERADORES && y || DE LA FORMA:

  comando1 && comando2 →SE EJECUTA *comando2* SI EL STATUS DE TERMINACIÓN DE *comando1* ES 0

  comando1 || comando2 →SE EJECUTA *comando2* SI EL STATUS DE TERMINACIÓN DE *comando1* ES ≠ 0

  comando1 && comando2 || comando3 →SE EJECUTA *comando2* SI EL STATUS DE TERMINACIÓN DE *comando1* ES 0 O *comando3* EN CASO CONTRARIO

Unix: Shell y Scripts

COMANDOS SÍNCRONOS Y ASÍNCRONOS

- LOS COMANDOS O GRUPOS DE COMANDOS SE EJECUTAN NORMALMENTE DE FORMA SÍNCRONA EN UN SHELL, ES DECIR, UNO TRAS OTRO, QUEDANDO EL SHELL EN ESPERA DE LA TERMINACIÓN.
- LOS COMANDOS O GRUPOS DE COMANDOS SEGUIDOS DEL OPERADOR (&) DE LA FORMA:

  comando & o (comando1; comando2; ….)&

  SE EJECUTAN DE FORMA ASÍNCRONA, QUEDANDO LIBRE EL SHELL PARA ATENDER CUALQUIER OTRA PETICIÓN.

Unix: Shell y Scripts

DEFINICIÓN DE VARIABLES

- UN SHELL PERMITE LA DEFINICIÓN DE VARIABLES
- UNA VARIABLE SIMPLE SE DEFINE MEDIANTE LA EXPRESIÓN:

  variable=[valor]

  DONDE, SI NO SE DA *valor*, LA VARIABLE TOMA EL VALOR *null*
- EL NOMBRE ES SENSIBLE A LAS MAYÚSCULAS Y MINÚSCULAS Y DEBE COMENZAR, PREFERIBLEMENTE, CON UN CARÁCTER ALFABÉTICO
- SE PUEDE DEFINIR UNA VARIABLE VECTOR MEDIANTE:

  declare –a variable

  A LA QUE SE PUEDE ASIGNAR VALORES MEDIANTE:

  variable[índice]=valor o variable=(valor1 valor2 ….)

Unix: Shell y Scripts

DEFINICIÓN DE VARIABLES

- REGLAS A TENER EN CUENTA AL ASIGNAR VALOR A UNA VARIABLE:
  - EL CARÁCTER *backslash* (\) ES EL CARÁCTER DE ESCAPE Y SIRVE PARA PRESERVAR EL VALOR LITERAL DEL CARÁCTER QUE SIGUE
  - LAS SIMPLES COMILLAS (') PRESERVAN EL VALOR LITERAL DE LOS CARACTERES ENTRE LAS MISMAS (MENOS LA COMILLA)
  - LAS DOBLES COMILLAS (") PRESERVAN EL VALOR LITERAL DE TODOS LOS CARACTERES EXCEPTO ($),(`) Y (\)

Unix: Shell y Scripts

DEFINICIÓN DE VARIABLES

- VALORES DE LA FORMA *$'string'* SON TRADUCIDOS A SU VALOR ANSI C STANDARD CORRESPONDIENTE DE ACUERDO CON LA SIGUIENTE TABLA

$'\a' →bell
$'\b'→*backspace*
$'\e'→*escape*
$'\f'→*form feed*
$'\n'→*newline*
$'\r'→*carriage return*
$'\t'→*horizontal tab*
$'\v'→*vertical tab*
$'\\'→*backslash*
$'\''→*single quote*
$'\nnn'→*octal nnn*
$'\xHH'→*hexa HH*
$'\cX'→*ctrl-X*

Unix: Shell y Scripts

EXPANSIÓN DE LAS VARIABLES

- LA FORMA MÁS SENCILLA DE RECUPERAR EL VALOR DE UNA VARIABLE SIMPLE ES *$variable*

  variable=pepe; echo $variable → *[xxx@triqui3 ~] $ pepe*
- SI SE QUIERE EVITAR CONFUSIONES CUANDO SE CONCATENAN TEXTOS, CONVIENE USAR *${variable}*

  echo ${variable}123 → *[xxx@triqui3 ~] $ pepe123*
- LOS VALORES DE LAS VARIABLES DE TIPO VECTOR SE RECUPERAN MEDIANTE *${variable[indice]}*

  declare –a vector; vector=(uno dos)

  echo ${vector[0]} ${vector[1]} → *[xxx@triqui3 ~] $ uno dos*

Unix: Shell y Scripts

EXPANSIÓN DE LAS VARIABLES

- LA EXPRESIÓN *${variable:-valor}* DEVUELVE EL VALOR DE LA VARIABLE SI EXISTE Y NO ES NULO. EN CASO CONTRARIO, *valor*

  variable=""; echo ${variable:-otro}→ *[xxx@triqui3 ~] $ otro*
- LA EXPRESIÓN *${variable:=valor}* DEVUELVE EL VALOR DE LA VARIABLE SI EXISTE Y NO ES NULO. EN CASO CONTRARIO, ASIGNA Y DEVUELVE *valor*
- LA EXPRESIÓN *${variable:?mensaje}* DEVUELVE EL VALOR DE LA VARIABLE SI EXISTE Y NO ES NULO. EN CASO CONTRARIO, MUESTRA EL MENSAJE EN LA SALIDA DE ERROR
- LA EXPRESIÓN *${variable:+valor}* DEVUELVE *valor* SI LA VARIABLE EXISTE Y NO ES NULA. EN CASO CONTRARIO, NO DEVUELVE NADA

Unix: Shell y Scripts

EXPANSIÓN DE LAS VARIABLES

- LA EXPRESIÓN *${variable:offset:n}* DEVUELVE *n* CARACTERES DEL VALOR DE LA VARIABLE, COMENZANDO POR LA POSICION *offset*

  variable="pepe perez"; echo ${variable:0:4} →*[xxx@triqui3 ~] $ pepe*

  SI SE OMITE *n*, DEVUELVE DESDE LA POSICIÓN offset HASTA EL FINAL

  echo ${variable:2:4} → *[xxx@triqui3 ~] $ pe perez*
- LA EXPRESIÓN *${!variable}* DEVUELVE EL VALOR DE LA VARIABLE CUYO NOMBRE ESTÁ CONTENIDO EN *variable:*

  var1=var2; var2=valor2; echo ${!var1} →*[xxx@triqui3 ~] $ valor2*

Unix: Shell y Scripts

EXPANSIÓN DE COMANDOS Y EXPRESIONES ARITMÉTICAS

- LA EXPRESIÓN *$(comando)* DEVUELVE LA SALIDA DEL COMANDO:

  echo $(ls) da el mismo resultado que ls

  Y LO MISMO SUCEDE CON `comando`:

  echo `ls` da el mismo resultado que ls
- LA EXPRESIÓN *$((expresión aritmética))* DEVUELVE EL VALOR DE LA EXPRESIÓN ARITMÉTICA, PERMITIENDO LA SUBSTITUCIÓN:

  i=0; echo $((i + 1)) → *[xxx@triqui3 ~] $ 1*

Unix: Shell y Scripts

EXPANSIÓN DE VARIABLES ESPECIALES

- *$0*: NOMBRE DE SHELL O SCRIPT
- *$1 … $n* : VARIABLES POSICIONALES (ARGUMENTOS PASADOS A UN SCRIPT O FUNCIÓN)
- $# : NÚMERO DE VARIABLES POSICIONALES
- $? : STATUS DE SALIDA DEL ÚLTIMO COMANDO
- $*, $@ : LISTA DE ARGUMENTOS PASADOS A UN SCRIPT O FUNCIÓN
- $$ PID DEL SHELL
- $! PID DEL ÚLTIMO PROCESO LANZADO EN *BACKGROUND*

Unix: Shell y Scripts

OTRAS VARIABLES PREDEFINIDAS

- EJECUTAR COMANDO *env*
- EJECUTAR *COMANDO set*

Unix: Shell y Scripts

SEÑALES / INTERRUPCIONES

- SON MENSAJES QUE RECIBEN LOS PROCESOS Y PROVOCAN UNA ATENCIÓN INMEDIATA
- PUEDEN TENER ORIGEN DIVERSO, POR EJEMPLO:
  - PULSACIÓN DE TECLADO (*ctrl-c*, *ctrl-d*, *ctrl-z*, …)
  - COMANDO *kill*
- ALGUNAS SEÑALES USUALES SON:
  - SIGEXIT (0) : TERMINAR EL SHELL (Ctrl-d)
  - SIGINT (2) : TERMINAR EL PROCESO EN CURSO (Ctrl-c)
  - SIGSTOP (24) : DETENER EL PROCESO EN CURSO (Ctrl-z)

Unix: Shell y Scripts

SEÑALES / INTERRUPCIONES

- LA ACCIÓN POR DEFECTO DE LAS INTERRUPCIONES SE PUEDEN ALTERAR MEDIANTE EL COMANDO:

  trap "comandos" señal
- PARA IGNORAR UNA SEÑAL, BASTA CON ESCRIBIR:

  trap "" señal
- Y, PARA RECUPERAR LA ACCIÓN POR DEFECTO:

  trap señal
- PARA ALGUNAS SEÑALES EXISTE UN PROCEDIMIENTO ALTERNATIVO:

  IGNOREEOF=n →trap "" SIGEXIT

Unix: Shell y Scripts

SHELLS Y SUBSHELLS

- CUANDO UN USUARIO ACCEDE AL SISTEMA, ARRANCA UN SHELL DENOMINADO "*shell de sesión*" QUE EXAMINA LOS COMANDOS Y LOS EJECUTA CUANDO LA SINTAXIS ES CORRECTA
- COMO YA SE HA VISTO, SE PUEDEN ARRANCAR NUEVOS SHELLS (*SUBSHELLS*) MEDIANTE EL COMANDO *bash* Y SALIR DE ELLOS CON EL COMANDO *BUILTIN exit*
- UN *SCRIPT* (FICHERO DE COMANDOS QUE TIENE PERMISO DE EJECUCIÓN) SE EJECUTA SIEMPRE EN UN SUBSHELL SALVO QUE SE PRECEDA CON UN PUNTO (.) SEGUIDO DE UN ESPACIO
- UN GRUPO DE COMANDOS ENTRE PARÉNTESIS ARRANCA UN SUBSHELL QUE TERMINA AUTOMÁTICAMENTE UNA VEZ EJECUTADOS LOS COMANDOS

Unix: Shell y Scripts

EL ENTORNO DE EJECUCIÓN

- TODO SHELL TIENE UN *entorno de ejecución* FORMADO POR:
  - Los ficheros abiertos al invocarlo
  - Las opciones habilitadas al invocarlo o habilitadas por el comando *shop*
  - El directorio actual
  - El modo de creación de ficheros definido por *umask*
  - Las respuestas a las interrupciones definidas por el comando *trap*
  - Variables definidas en él o heredadas del shell padre mediante el comando *export*
  - Las funciones definidas en él o heredadas del shell padre mediante el comando *export -f*
  - Los alias de los comandos definidos mediante el comando *alias*

Unix: Shell y Scripts

EL ENTORNO DE EJECUCIÓN

- UN COMANDO QUE NO SEA *BUILTIN* O FUNCIÓN DEL SHELL SE EJECUTA EN UN ENTORNO SEPARADO FORMADO POR :
  - Los ficheros abiertos al invocar el shell
  - El directorio actual
  - El modo de creación de ficheros definido por *umask*
  - Las variables y funciones del shell marcadas como exportadas y las variables pasadas al comando como parámetros
  - Las respuestas a las interrupciones heredadas por el shell

  Y LOS CAMBIOS REALIZADOS EN ESTE ENTORNO NO AFECTAN AL ENTORNO DEL SHELL PADRE
- LOS GRUPOS DE COMANDOS ENTRE PARÉNTESIS Y LOS COMANDOS ASÍNCRONOS SE EJECUTAN EN UN ENTORNO COPIA DEL SHELL PADRE, PERO LOS CAMBIOS QUE SE REALICEN NO LE AFECTAN.